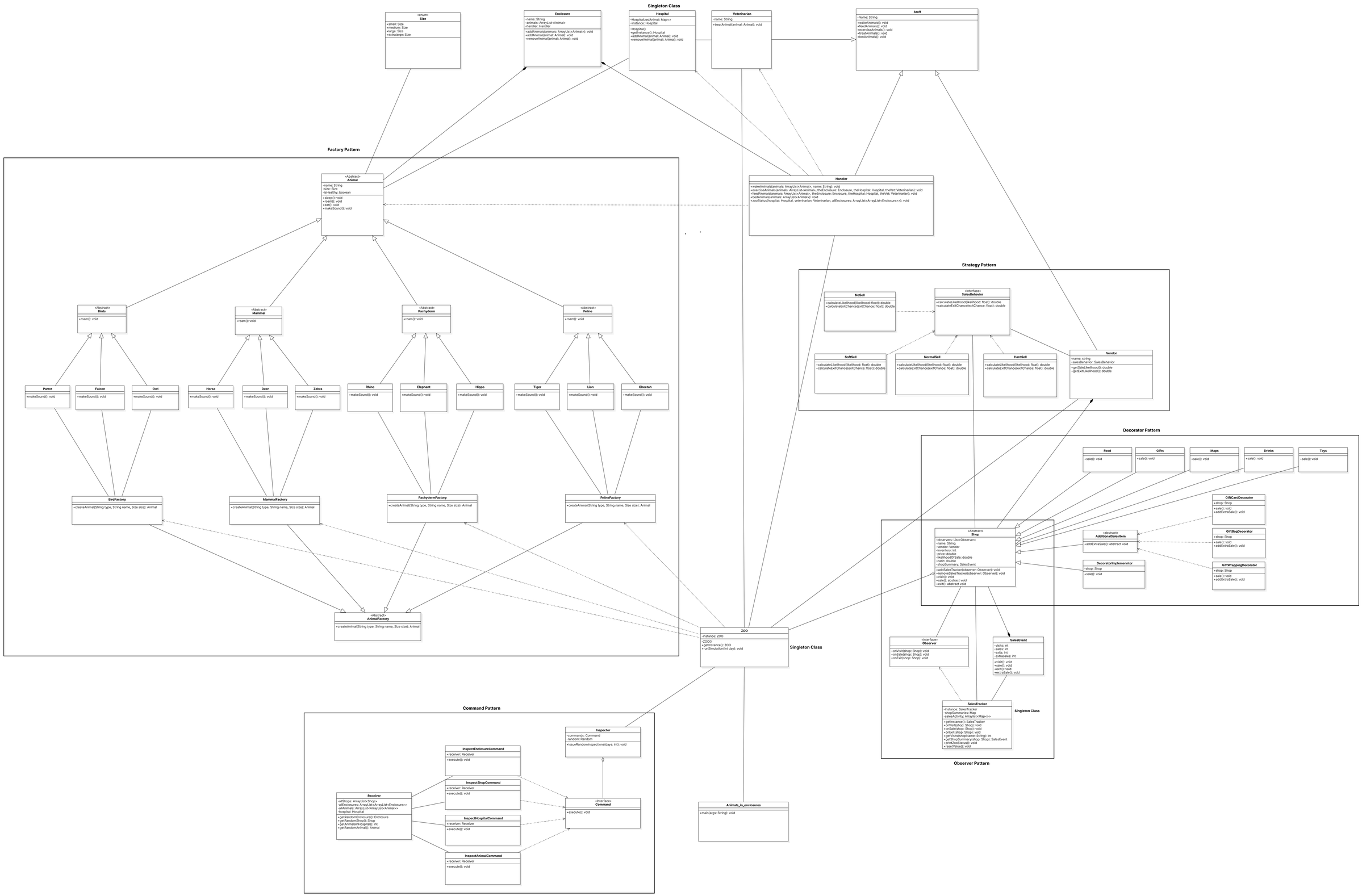
«enum»
Size
+small: Size
+medium: Size
+large: Size
+extralarge: Size
Enclosure
-name: String
-animals: ArrayList<Animal>
-handler: Handler
+addAnimals(animals: ArrayList<Animal>): void
+addAnimal(animal: Animal): void
+removeAnimal(animal: Animal): void
Singleton Class
Hospital
-HospitalizedAnimal: Map<>
-instance: Hospital
-Hospital()
+getInstance(): Hospital
+addAnimal(animal: Animal): void
+removeAnimal(animal: Animal): void
Veterinarian
-name: String
+treatAnimal(animal: Animal): void
Staff
-Name: String
+wakeAnimals(): void
+feedAnimals(): void
+exerciseAnimals(): void
+treatAnimals(): void
+bedAnimals(): void
Factory Pattern
«Abstract»
Animal
+name: String
-size: Size
-isHealthy: boolean
+sleep(): void
+roam(): void
+eat(): void
+makeSound(): void
Handler
+wakeAnimals(animals: ArrayList<Animal>, name: String): void
+exerciseAnimals(animals: ArrayList<Animal>, theEnclosure: Enclosure, theHospital: Hospital, theVet: Veterinarian): void
+feedAnimals(animals: ArrayList<Animal>, theEnclosure: Enclosure, theHospital: Hospital, theVet: Veterinarian): void
+bedAnimals(animals: ArrayList<Animal>): void
+zooStatus(hospital: Hospital, veterinarian: Veterinarian, allEnclosures: ArrayList<ArrayList<Enclosure>>): void
«Abstract»
Birds
+roam(): void
«Abstract»
Mammal
+roam(): void
«Abstract»
Pachyderm
+roam(): void
«Abstract»
Feline
+roam(): void
Parrot
+makeSound(): void
Falcon
+makeSound(): void
Owl
+makeSound(): void
Horse
+makeSound(): void
Deer
+makeSound(): void
Zebra
+makeSound(): void
Rhino
+makeSound(): void
Elephant
+makeSound(): void
Hippo
+makeSound(): void
Tiger
+makeSound(): void
Lion
+makeSound(): void
Cheetah
+makeSound(): void
BirdFactory
+createAnimal(String type, String name, Size size): Animal
MammalFactory
+createAnimal(String type, String name, Size size): Animal
PachydermFactory
+createAnimal(String type, String name, Size size): Animal
FelineFactory
+createAnimal(String type, String name, Size size): Animal
«Abstract»
AnimalFactory
+createAnimal(String type, String name, Size size): Animal
Strategy Pattern
NoSell
+calculateLikelihood(likelihood: float): double
+calculateExitChance(exitChance: float): double
«Interface»
SalesBehavior
+calculateLikelihood(likelihood: float): double
+calculateExitChance(exitChance: float): double
SoftSell
+calculateLikelihood(likelihood: float): double
+calculateExitChance(exitChance: float): double
NormalSell
+calculateLikelihood(likelihood: float): double
+calculateExitChance(exitChance: float): double
HardSell
+calculateLikelihood(likelihood: float): double
+calculateExitChance(exitChance: float): double
Vendor
-name: string
-salesBehavior: SalesBehavior
+getSaleLikelihood(): double
+getExitLikelihood(): double
Decorator Pattern
Food
+sale(): void
Gifts
+sale(): void
Maps
+sale(): void
Drinks
+sale(): void
Toys
+sale(): void
GiftCardDecorator
+shop: Shop
+sale(): void
+addExtraSale(): void
«abstract»
AdditionalSalesItem
+addExtraSale(): abstract void
GiftBagDecorator
+shop: Shop
+sale(): void
+addExtraSale(): void
DecoratorImplementor
-shop: Shop
+sale(): void
GiftWrappingDecorator
+shop: Shop
+sale(): void
+addExtraSale(): void
«Abstract»
Shop
-observers: List<Observer>
-name: String
-vendor: Vendor
-inventory: int
-totalSales: int
-totalLoadOfSale: double
-cash: double
-shopSummary: SalesEvent
+addSalesTracker(observer: Observer): void
+removeSalesTracker(observer: Observer): void
+visit(): void
+sale(): abstract void
+exit(): abstract void
«Interface»
Observer
+onVisit(shop: Shop): void
+onSale(shop: Shop): void
+onExit(shop: Shop): void
SalesEvent
-visits: int
-sales: int
-exits: int
-extrasales: int
+visit(): void
+sale(): void
+exit(): void
+extraSale(): void
SalesTracker
-instance: SalesTracker
-shopSummaries: Map
-salesActivity: ArrayList<Map<>>
+getInstance(): SalesTracker
+onVisit(shop: Shop): void
+onSale(shop: Shop): void
+onExit(shop: Shop): void
+getVisits(shopName: String): int
+getShopSummary(shop: Shop): SalesEvent
+printZooStatus(): void
+resetValue(): void
Singleton Class
Observer Pattern
ZOO
-instance: ZOO
-ZOO()
+getInstance(): ZOO
+runSimulation(int day): void
Singleton Class
Command Pattern
Inspector
-commands: Command
-random: Random
+issueRandomInspections(days: int): void
InspectEnclosureCommand
+receiver: Receiver
+execute(): void
InspectShopCommand
+receiver: Receiver
+execute(): void
InspectHospitalCommand
+receiver: Receiver
+execute(): void
InspectAnimalCommand
+receiver: Receiver
+execute(): void
Receiver
-allShops: ArrayList<Shop>
-allEnclosures: ArrayList<ArrayList<Enclosure>>
-allAnimals: ArrayList<ArrayList<Animal>>
-Hospital: Hospital
+getRandomEnclosure(): Enclosure
+getRandomShop(): Shop
+getAnimalsInHospital(): int
+getRanDomAnimal(): Animal
«Interface»
Command
+execute(): void
Animals_in_enclosures
+main(args: String): void